Scribe: Florian Milanovici and Antonio Montalban Lecturer: Andrew Myers

1 Substitution

We have already defined the β-reduction:

$$(\lambda x\, e_0)e_1 \mapsto_\beta e_0\{e_1/x\}$$

But we have not defined how to do the substitution, which is less straightforward than it seems. Many mathematicians like Newton, Church and Gödel tried to define substitution but they all failed to give a correct definition. It was around 1950 when the first right definition appeared. There are some things that one has to take care of when defining substitution, for example if we want to do the following substitution:

$$(x(\lambda x\, x))\{y/x\}$$

we cannot replace every occurrence of the variable x by y. We have to replace y for x only when x appears free and get

$$y(\lambda x\, x).$$

Another problem is *Variable Capture*. For example, if we have

$$y(\lambda x\, (x\, y))\{x/y\}$$

and we just substitute x for y we would get

$$x(\lambda x\, (x\, x))$$

but this is not correct because the last occurrence of x has been captured by the lambda expression. So, an expression should not be substituted into a context when some of its free variables would become bound in that context.

We start defining $\mathtt{FV}[\![e]\!]$, the set of free variables of e. The definition is by recursion on the structure of the λ-term e:

$$\begin{array}{lcl} \mathtt{FV}[\![x]\!] & = & \{x\} \\ \mathtt{FV}[\![e_0 e_1]\!] & = & \mathtt{FV}[\![e_0]\!] \cup \mathtt{FV}[\![e_1]\!] \\ \mathtt{FV}[\![\lambda x\, e]\!] & = & \mathtt{FV}[\![e]\!] - \{x\} \end{array}$$

Now we define substitution. We define $\{e/x\}$ as a postfix function by recursion on the structure of the λ-terms:

$$\begin{array}{lcll} x\{e/x\} & = & e & \\ y\{e/x\} & = & y & \text{where } y \neq x \\ e_0 e_1\{e/x\} & = & (e_0\{e/x\})(e_1\{e/x\}) & \\ \lambda x\, e_0\{e/x\} & = & \lambda x\, e_0 & \\ \lambda y\, e_0\{e/x\} & = & \lambda y\, (e_0\{e/x\}) & \text{where } y \neq x \wedge y \notin \mathtt{FV}[\![e]\!] \\ \lambda y\, e_0\{e/x\} & = & \lambda y'\, (e_o\{y'/y\}\{e/x\}) & \text{where } y \neq x \wedge y \in \mathtt{FV}[\![e]\!] \wedge \\ & & & y' \notin \mathtt{FV}[\![e_0]\!] \cup \mathtt{FV}[\![e]\!] \cup \{x\} \end{array}$$

In the last line of the definition we first change the variable y, in $\lambda y\, e_0$, by a fresh variable y', and then we do the substitution. In this way no free variable of e is captured. It is clear that when we transform $\lambda y\, e_0$ into $\lambda y'\, (e_0\{y'/y\})$ we are not changing the meaning of the term. This is because in some sense those terms are the same. This become clear when we draw a Stoy Diagram. In the *Stoy Diagram* of a λ-term we do not write the names of the variables, we just draw points and we link each point to the λ which is bounding it. Figure 1 has an example of a Stoy Diagram.

The meaning of a lambda expression does not depend on the name of the argument variable:

$$(\lambda x\, x) \quad = \quad (\lambda x'\, x')$$

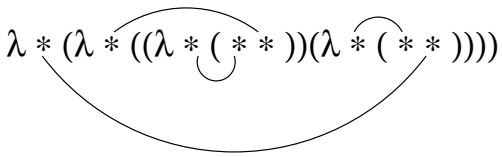

Figure 1: Stoy Diagram of the term $\lambda p(\lambda q(\lambda\,(pq))(\lambda r\,(pr)))$

This brings us to the process of α-*reduction* (or α-renaming), where a function argument is renamed yielding an equivalent expression:

$$(\lambda x\,e) \mapsto_\alpha (\lambda x'\,e\{x'/x\}) \quad \text{where } x' \notin \texttt{FV}[\![e]\!]$$

Two lambda expressions e and e' are α-*equivalent* if they can be converted to each other using α-reductions (or equivalently if they have the same Stoy diagram). We have the following axioms and rules for α-equivalence:

$$\frac{}{x \approx_\alpha x}$$

$$\frac{}{(\lambda x\,e) \approx_\alpha (\lambda x'\,e\{x'/x\})} \quad \text{where } x' \notin \texttt{FV}[\![e]\!]$$

$$\frac{e_0 \approx_\alpha e_0' \quad e_1 \approx_\alpha e_1'}{e_0\ e_1 \approx_\alpha e_0'\ e_1'}$$

$$\frac{e \approx_\alpha e'}{(\lambda x\,e) \approx_\alpha (\lambda x\,e')}$$

So far we've seen two reductions of lambda calculus; namely, the α and the β reductions:

$$\begin{array}{ll}(\lambda x\,e) \mapsto_\alpha (\lambda x'\,e\{x'/x\}) & \text{where } x' \notin \texttt{FV}[\![e]\!]\\ (\lambda x\,e_0)e_1 \mapsto_\beta e_0\{e_1/x\} & \end{array}$$

We will introduce now the third reduction in lambda calculus, the η-reduction. We said that terms represents functions, so two terms should be equal if they represent the same function by extension. i.e.:

$$\text{if } f\ x = g\ x \text{ then } f = g$$

For example the two functions, $(\lambda x\,(ex))$ and e, are equal by extension if x is not free in e. If we apply them to the same argument e' the result that is obtained in both cases is ee'. Thus we can η-reduce $(\lambda x\,(e\,x))$ to e:

$$(\lambda x\,(e\,x)) \mapsto_\eta e$$

So the possible reductions on λ-terms are :

$$\begin{array}{ll}(\lambda x\,e) \mapsto_\alpha (\lambda x'\,e\{x'/x\}) & \text{where } x' \notin \texttt{FV}[\![e]\!]\\ (\lambda x\,e_0)e_1 \mapsto_\beta e_0\{e_1/x\} & \\ (\lambda x\,(e\,x)) \mapsto_\eta e & \text{if } x \notin \texttt{FV}[\![e]\!]\end{array}$$

A reducible expression is commonly referred to as a *redex*. A λ-term is said to be in *normal form* if no reductions can be performed on it or on any of its subexpressions. We note that not all lambda expressions have a normal form. For example $\Omega = (\lambda x\,(xx))(\lambda x\,(xx))$ is never going to reach normal form.

Order of reductions:

1 **Normal order**: β or η reductions are applied on the leftmost lambda subterm until no reductions can be applied. This is equivalent to *call by name* evaluation. Normal order always finds a normal form if there is one.

2 **Applicative order**: Only apply β-reduction if the argument is in normal form. It corresponds to the call-by-name evaluation. It does not always find a normal form even if there is one.

3 **Nondeterministic evaluation**: There is no predefined order of evaluation:

$$\frac{e_0 \mapsto e_0'}{e_0\ e_1 \mapsto e_0'\ e_1}$$

$$\frac{e_1 \mapsto e_1'}{e_0\ e_1 \mapsto e_0\ e_1'}$$

Can we get different normal forms by evaluating in different orders? The answer is no and it is given by the Church-Rosser theorem that states that nondeterministic evaluation of an expression does not result in nondeterminism of the reduced normal form.

Church-Rosser Theorem: $\forall e_0, e_1, e_2.\ e_0 \mapsto^* e_1\ \wedge\ e_0 \mapsto^* e_2 \Rightarrow \exists e_3.\ e_1 \mapsto^* e_3\ \wedge\ e_2 \mapsto^* e_3$

Any transition relation that satisfies this theorem is said to have the *diamond property* or *confluence*.